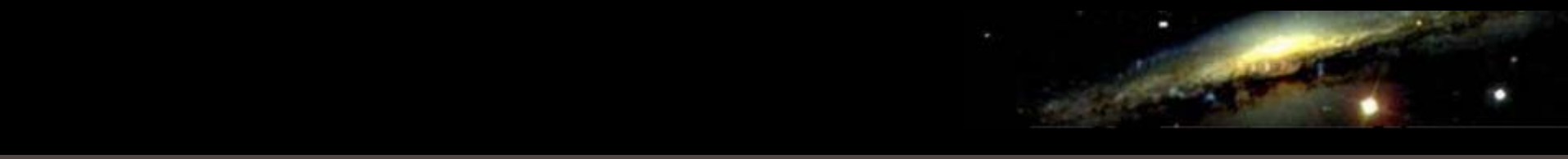


Measuring Database Usage and Data Mining Performance in the Sloan Digital Sky Survey

Ani Thakar

Alex Szalay, Maria Nieto-Santisteban,

Wil O'Mullane, George Fekete, Nolan Li

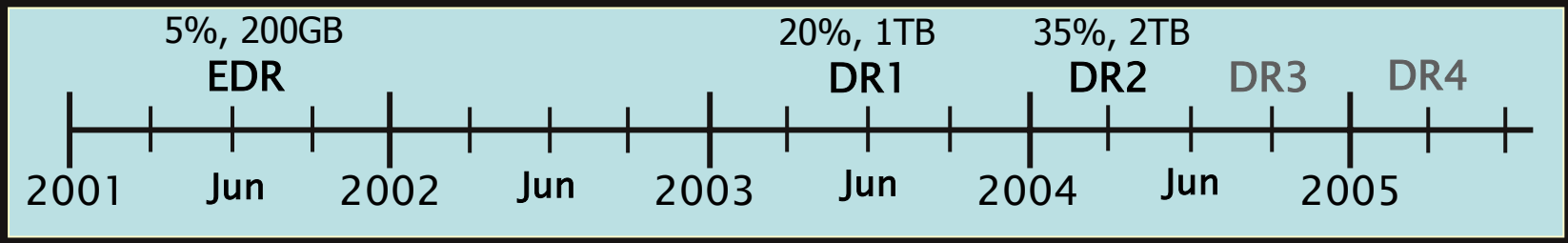
The Johns Hopkins University

Jim Gray

Microsoft Research

SDSS

SDSS Overview



- Online access via SkyServer site skyserver.sdss.org
- DR2 released March 15, 2004
  - 2 TB of catalog data served online, 90M objs, 300k spectra
- DR1 released June 2003
  - Mirrored at several sites internationally
    - N America – SDSC, CADC (Victoria), UPitt
    - Europe – UK (Edinburgh), Max Planck
    - Asia – Japan, India
  - VO services (SkyQuery, SIAP, Spectra)

Measuring SDSS Performance

- Towards Grid computing with databases
  - Application to astronomy domain
  - Using SDSS as a testbed
- SkyServer logging
  - Extensive logging of web and DB traffic
  - Web usage, DB usage, DB performance
  - SkyServer Traffic Site
- Scalable data access
  - We are doing several things
  - Reliable high-speed data access for Grid applications
  - Benchmarking database vs file access
- Logging for the VO

Benefits of Logging

- Usage/traffic profiles great management resource
  - Invaluable for funding proposals and reviews
    - Are we meeting user requirements?
  - Monitor impact of press articles
- Track errors on web pages
- Use data for load balancing
- Find crawlers and “inconsiderate” users
- Track server performance
- Guide schema and interface design
  - How often are SDSS photo flags being used?
  - Are people getting clean photometry?
  - Are they filtering out invalid data values?
  - How comfortable are people with SQL/complex queries?
- Help prospective mirror sites to budget hardware

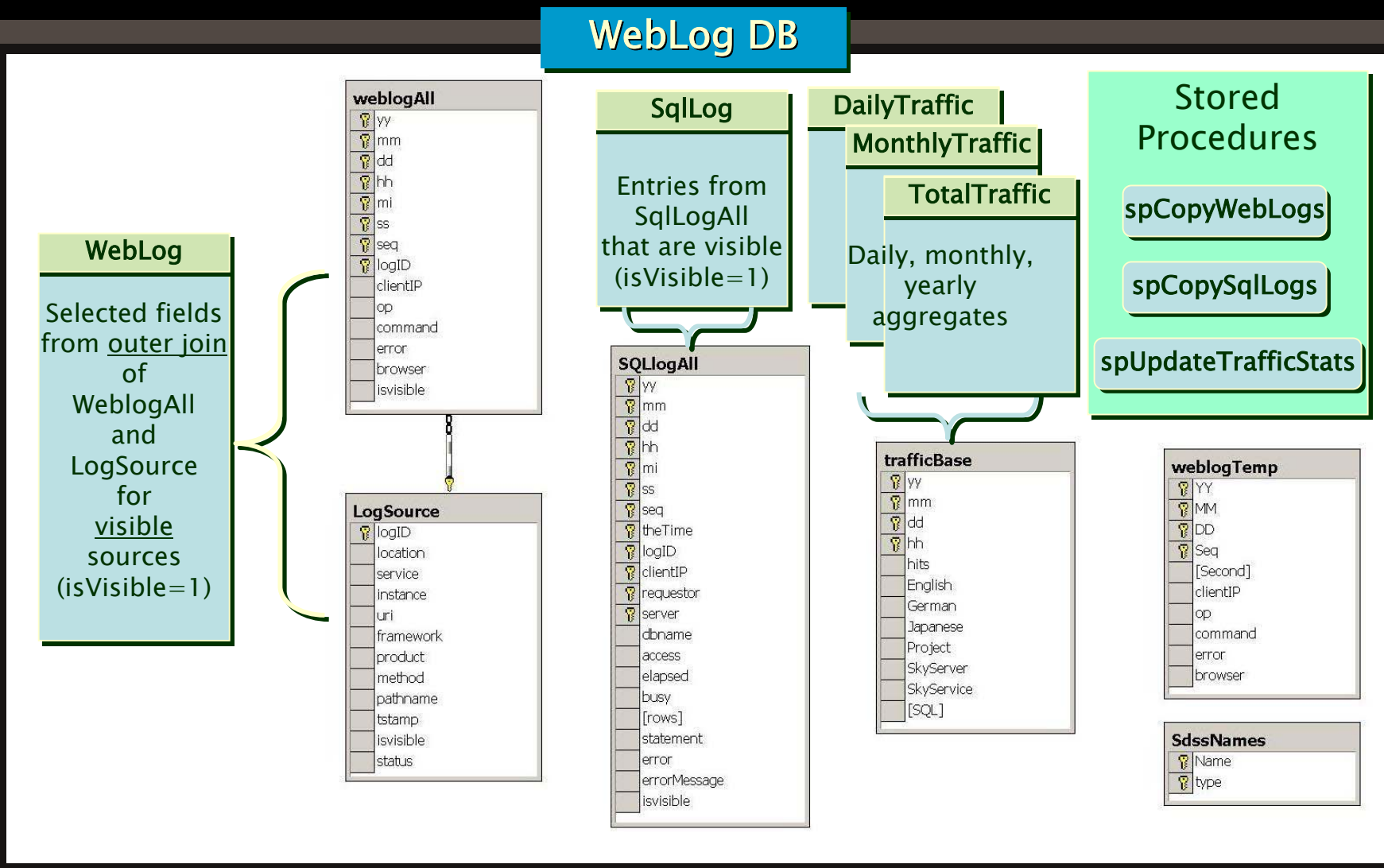
What is logged currently

- Web hits (IIS W3C log format)
- SQL queries
  - skyServer (browser, sdssQA, emacs, sqlcl)
  - CasJobs batch query
- Both collected from multiple sources
  - JHU
    - 1 Web server, 2 DB servers)
    - Soon 3-way webserver cluster, 2 DB servers
    - DR1 and DR2
  - FNAL
    - DR2: 2 Web servers, 2 DB servers (collab & public)
    - Soon DR3: 1 more web server, 2 more DB servers
- In the future, mirror sites around the world

Log Harvesting

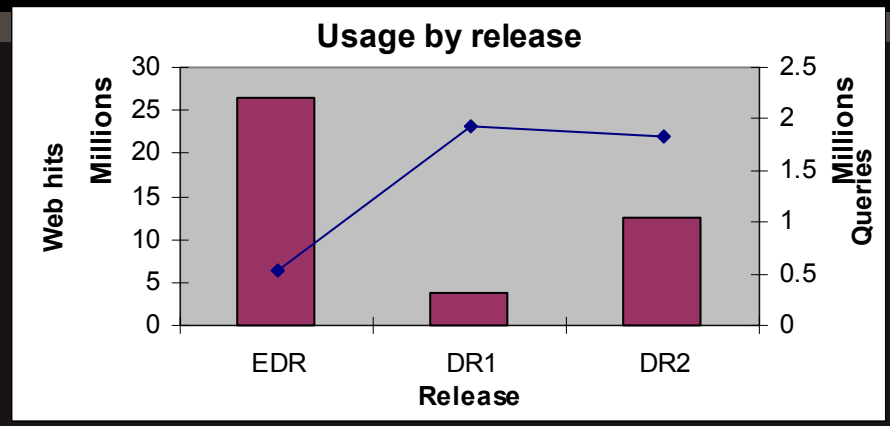
- Logs from all sources harvested at JHU
- Log server copies local web and SQL logs
  - Weblogs copied with xcopy, bcp-ed into DB
  - Local SQL logs collected with bulk inserts
- Remote log harvesting via WebServices
  - Currently using CasJobs to deliver FNAL logs
  - Future – special purpose web service
- Log server runs hourly update
  - Appends all entries since previous update
- All logs harvested into Weblog DB

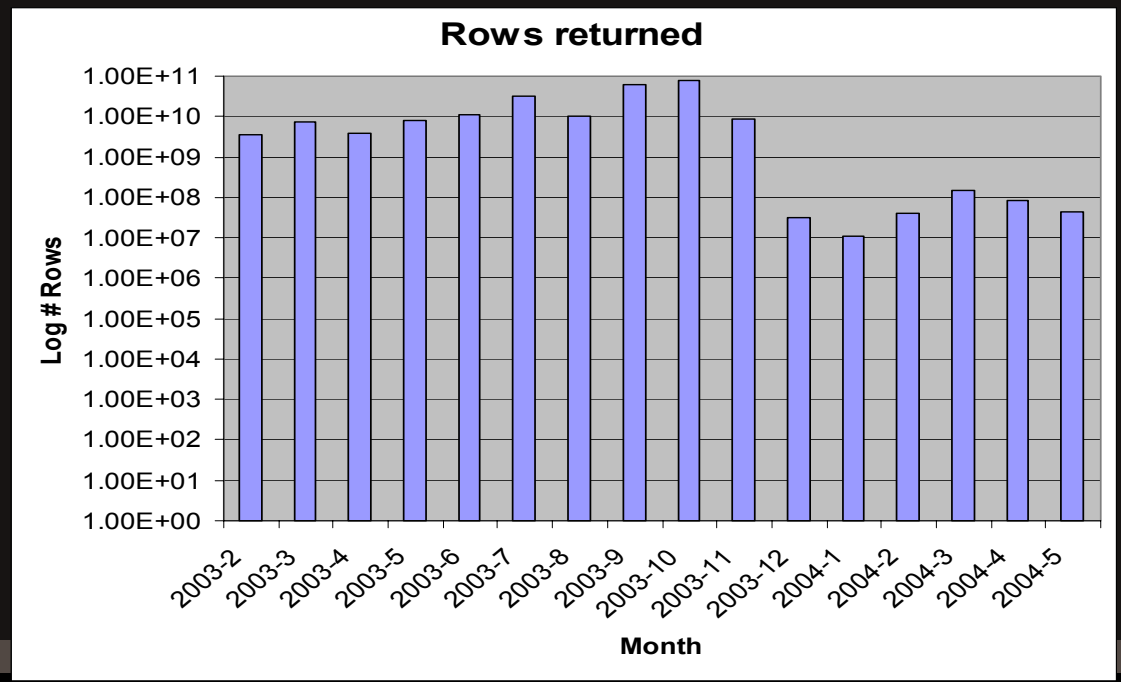
Log Harvesting Data Model



SkyServer Usage Data

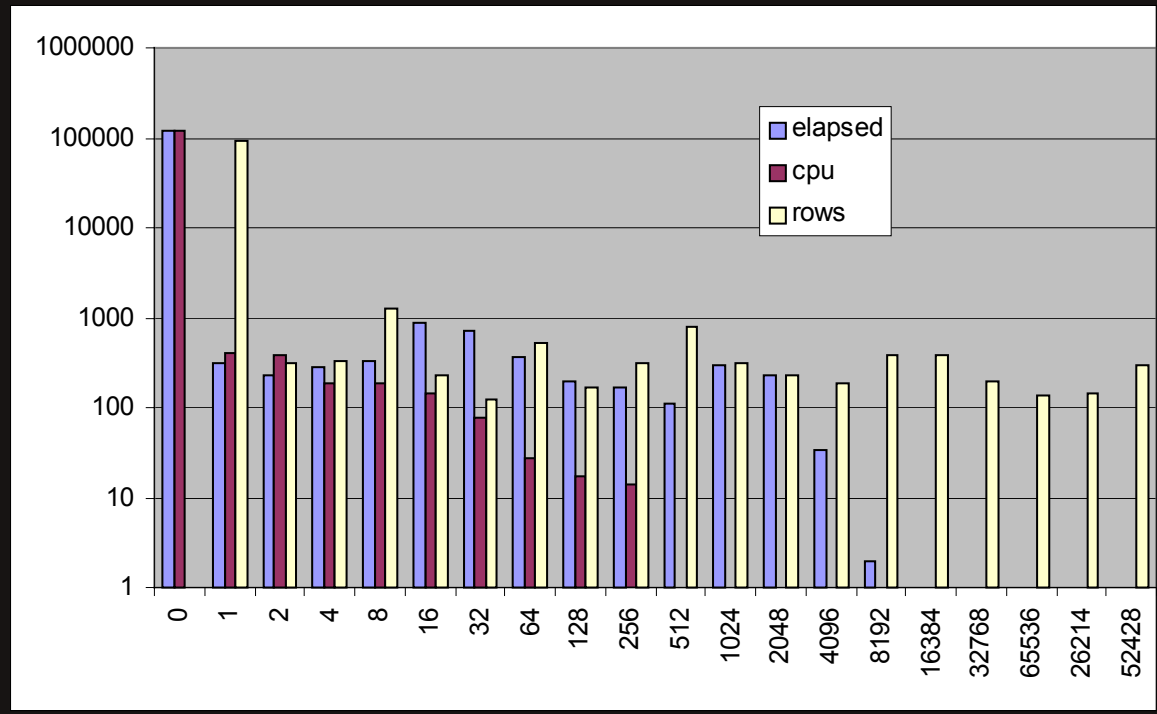
Error	#Hits
301	32520
302	934564
304	5027736
403	15099
404	410605
405	3208
411	974
502	26
503	19
200	35779141
206	64524
207	4501
400	1845
401	145047
406	3297
414	3
416	544
500	227732
501	376





SkyServer Usage Patterns

- 2 years of EDR +
- 6 months of DR1
- Access patterns:
  - Lots of small users, instant response
  - 1/f distribution of request sizes (tail of the lognormal)
- How to make everybody happy?



We need

- A separate interactive and batch server
- Access to full SQL with extensions
- Browser as well as SOAP access

Scalable Data Access

- Spatial Index (HTM) built into SQL database
- Vertical partitioning, precomputed joins,neighbors
- Data striping (filegroups on disk)
- Batch vs Interactive DB servers
  - Motivated by perf data from EDR, DR1
    - Vast majority of queries are quick (under 1 min)
    - Few slow queries bog down system for everyone
  - Batch query service CasJobs (O'Mullane et al., 2003)
    - Hosted on separate server from interactive service
    - Queries unlimited in time and (output size)
    - Personal user space on batch server (MyDB)
    - Ability to share query results with collaborators

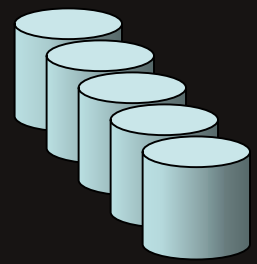
High-speed Grid data access

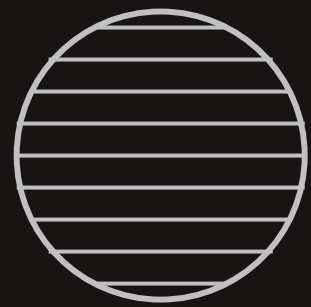
- Horizontal data partitioning across cluster
  - Parallel access to data
  - Spatially partition data using declination *zones*
    - Zones are a quick way to do spatial decomposition inside SQL, no spatial indexing required
  - Incorporate partitioning into data import workflow system (sqlLoader)
- Benchmark cluster finding algorithm
  - Compare file-based TAM/Grid implementation with SQL implementation in SDSS database
- Megajoins and data Ferris Wheel

Partitioning Strategy

- Two-Step Process

1. Distribute data homogenously among servers.
   - Each server has roughly the same amount of objects.
   - Objects inside servers are spatially related.
   - Balances the workload among servers.
   - Queries redirected to the server holding the data.
   - Each server add a buffer region on top and bottom (First and Last server hold only one buffer)
2. Applies the Cluster finding algorithm as described

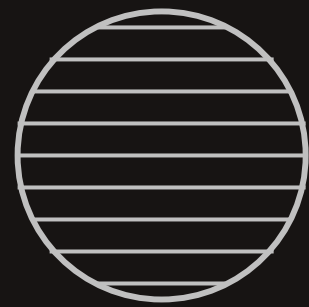




Mapping the Sphere into Zones

- Each Zone is a declination stripe of height h.
- In principle, h can be any number. In practice, 30 arcsec. (DR1 => 8000 ZONES)
- South-pole zone = Zone 0.
- Each object belongs to one Zone:

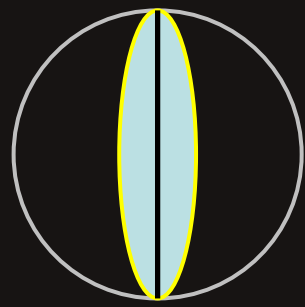
$$\text{ZoneID} = \text{floor}\ (\ (\text{dec} + 90)\ /\ h\ )$$



- Each server holds N "contiguous" Zones.
- N is determined by the number of objects that each Zone contains and the number of servers in the cluster.
  - Not all servers contain the same number of zones.
  - Not all servers cover the same declination range.

Margins & Buffers

- To improve queries around Ra = 0 (or 360):
  - Duplicate objects inside Ra = [-1, 0) and Ra = (360, 361] facilitates searches.
  - Objects in the margin area are marked as Wrap.



- To guarantee that neighboring searches can be fully satisfied inside a single server some zones are replicated:
  - Each server adds 2 extra buffer regions of height R_M
  - R_M, is the maximum neighboring distance we assume (1 degree).
  - Objects in buffers are "marked" as Visitors to the Server.

Cluster Finding Algorithm

- Five main steps (Annis et al. 2002)*
  1. fieldPrep: Extracts from the main data set the required measurement of interest. <- List of galaxies.
  2. brgSearch: Calculates the unweighted BCG likelihood for each galaxy (unweighted by galaxy count) <- Main filtering step.
  3. bcgSearch: Calculates the weighted BCG likelihood. <- Looks for neighbors with similar characteristics. (Max Radius=0.5 deg).
  4. bcgCoalesce: Determines whether a galaxy is the most likely galaxy in the neighborhood to be the center of the cluster.
  5. getCatalog: Removes suspicious results and produces and stores the final cluster catalog.

* *Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey ,* in Proc. of Super Computing 2002

Finding clusters in TAM

(TAM = Terabyte Analysis Machine, Annis et al.)

A

1. fieldPrep: Creates 2 files, one 0.5 x 0.5 deg^2 TARGET and one 1x1 deg^2 BUFFER (ideally the buffer should be 1.5 x 1.5 deg^2, but cannot afford)

T ~ 3.5k obj/0.25deg^2 B ~ 16k obj/1deg^2

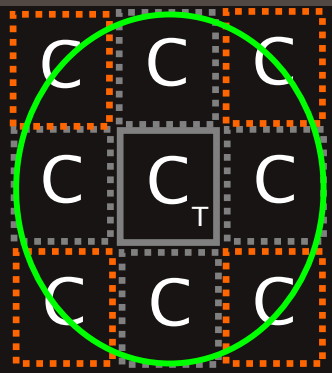
2. brgSearch: For each Galaxy in T calculates the BCG likelihood, chisquare$_1$ (unweighted by galaxy count) filtering out those with chisquare$_1$ < 7

3. bcgSearch: Counts the number of neighbors with similar characteristics and weights the BCG likelihood. The search radius depends on z.

chisquare = LOG (ngal +1) – chisquare$_1$

Step A: Generates a list of candidates and stores it in file C_T. About 2.5-4% of T are candidates.

Finding clusters in TAM

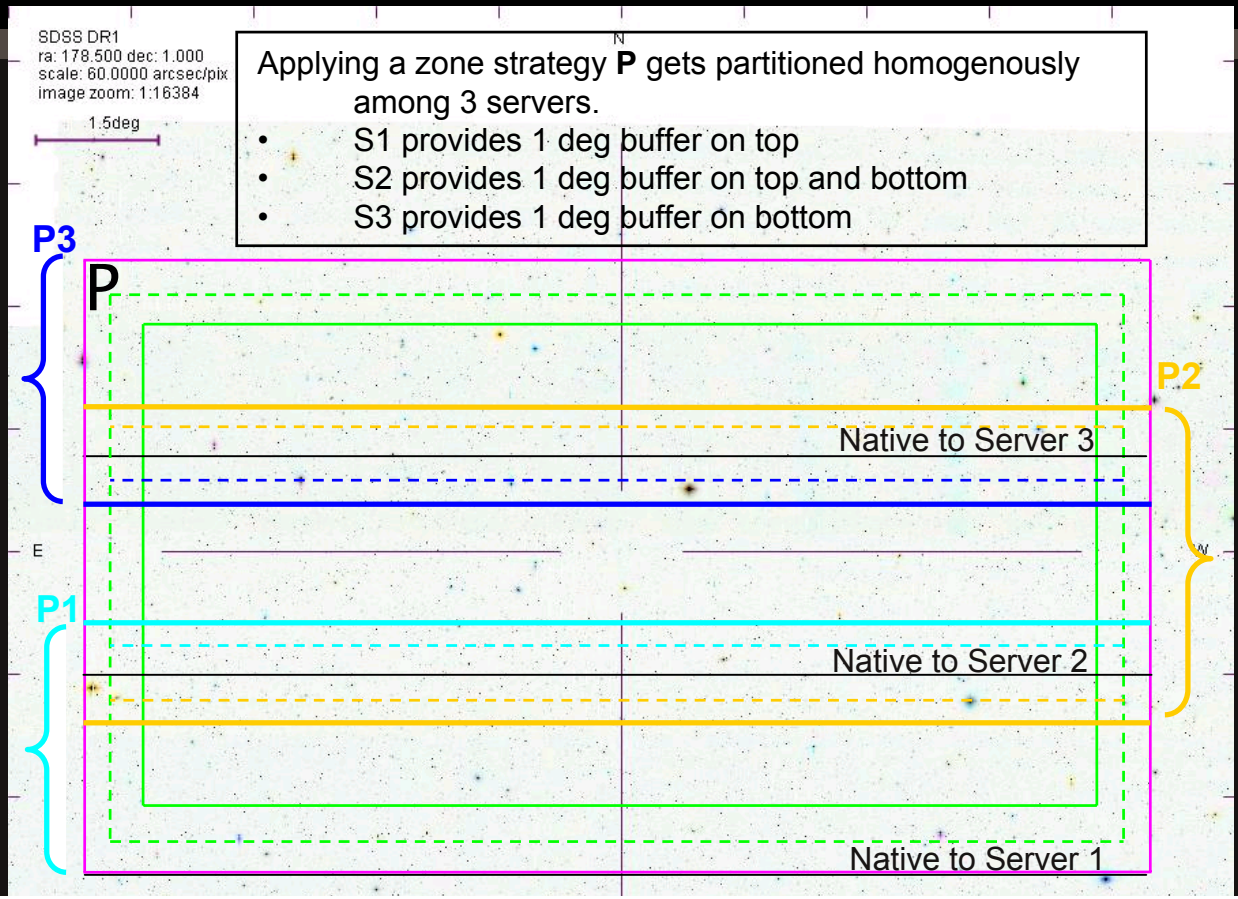


B

4. bcgCoalesce: For each galaxy g in C_T, this task determines whether or not g is the most likely to be the center of the cluster among other candidates in the neighborhood. The search radius varies depending on the galaxy redshift. The surrounding C files provide a radius close to 0.5 deg for candidates near to the border. Orange squares are actually not included due to performance issues.
5. getCatalog: Some times, there is no buffer data which implies the results are compromised. This data is removed from the final cluster catalog.

Step B: Finds ~ 4.5 clusters / 0.25 deg^2

SQL Server: Finding Candidates

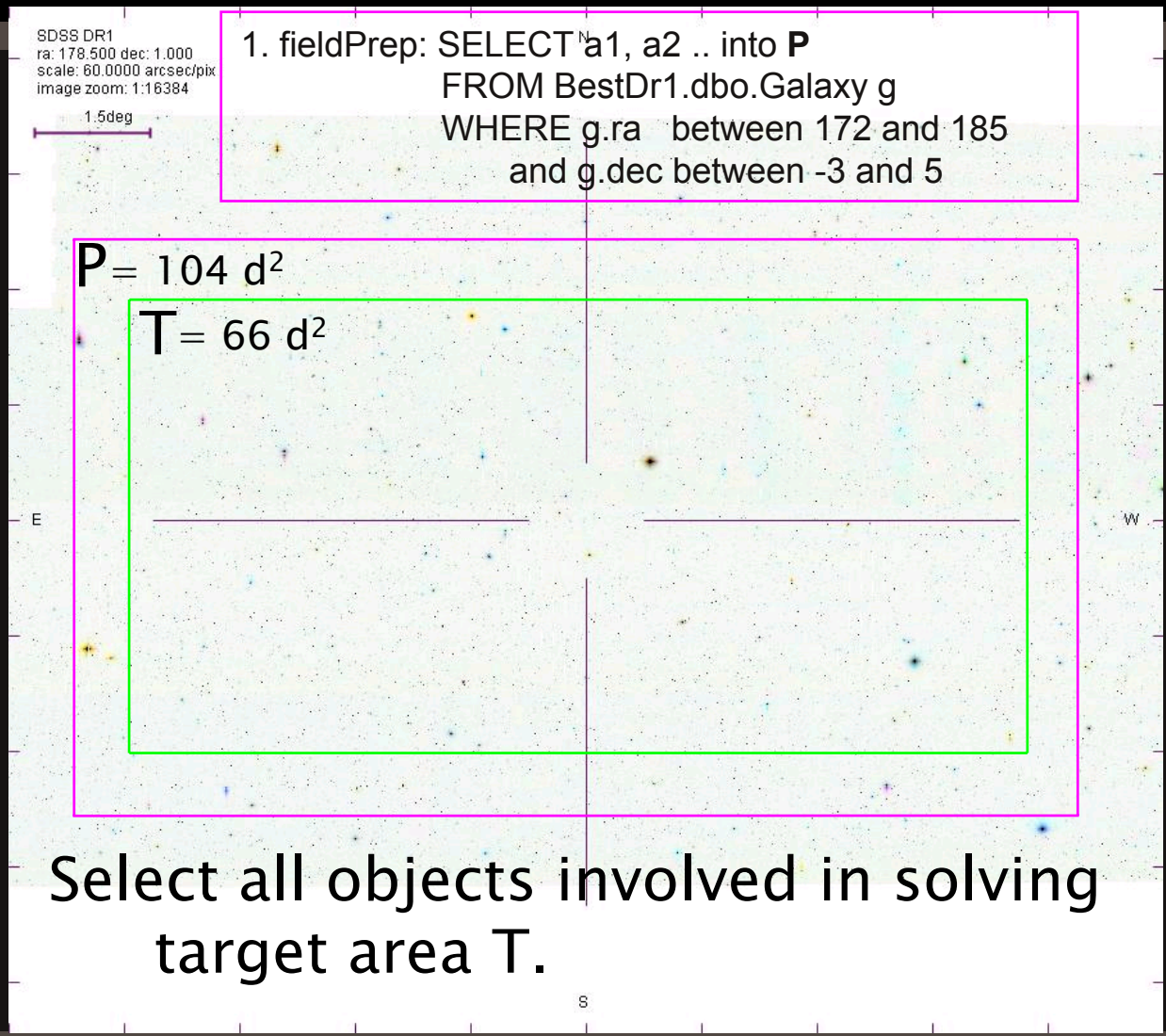


Total duplicated data = 4 x 13 deg^2. Total duplicated work = (1 object processed more than once) = 2 x 11 deg^2

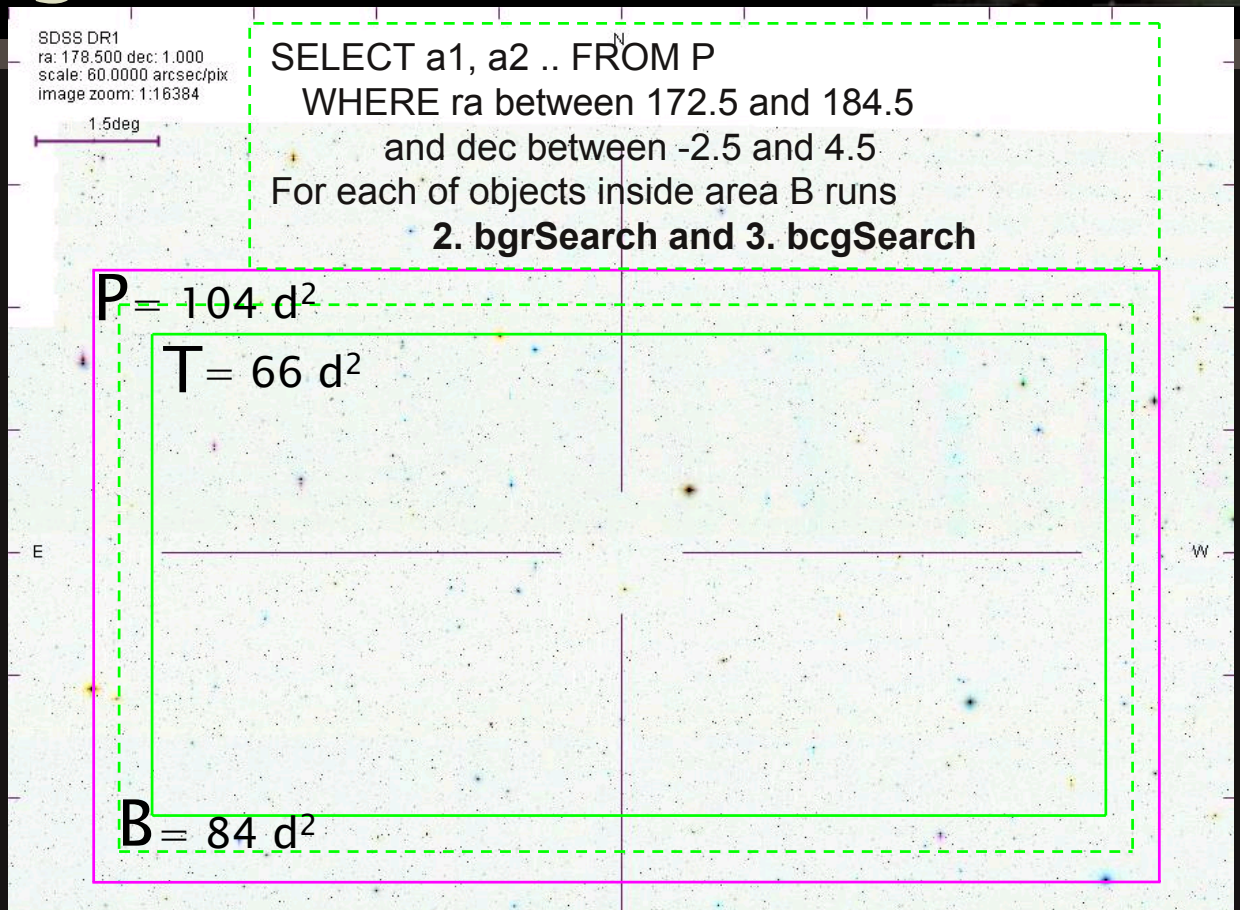
Maximum time spent by the thicker partition=2h 15' (other 2servers ~ 1h 50')

Processing 66 deg^2 using 10 nodes on TAM = ~ 7 hours

Finding clusters in SQL Server



Finding clusters in SQL Server



All objects .5 deg away of T are inspected to decide whether they are Candidates. Searches for neighbors include objects from **P**. More accurately computed because **P** provides a whole 0.5 degree. (TAM could afford only 0.25). This step creates a **Candidates** table, **C**.

Finding clusters in SQL Server

All candidates inside the Target area are inspected to decide whether or not they are the center of the cluster. Each candidate counts Candidate neighbors in C which guarantees a whole 0.5 degree of candidates properly computed. This step creates the **Clusters** table.

SQL Server vs TAM

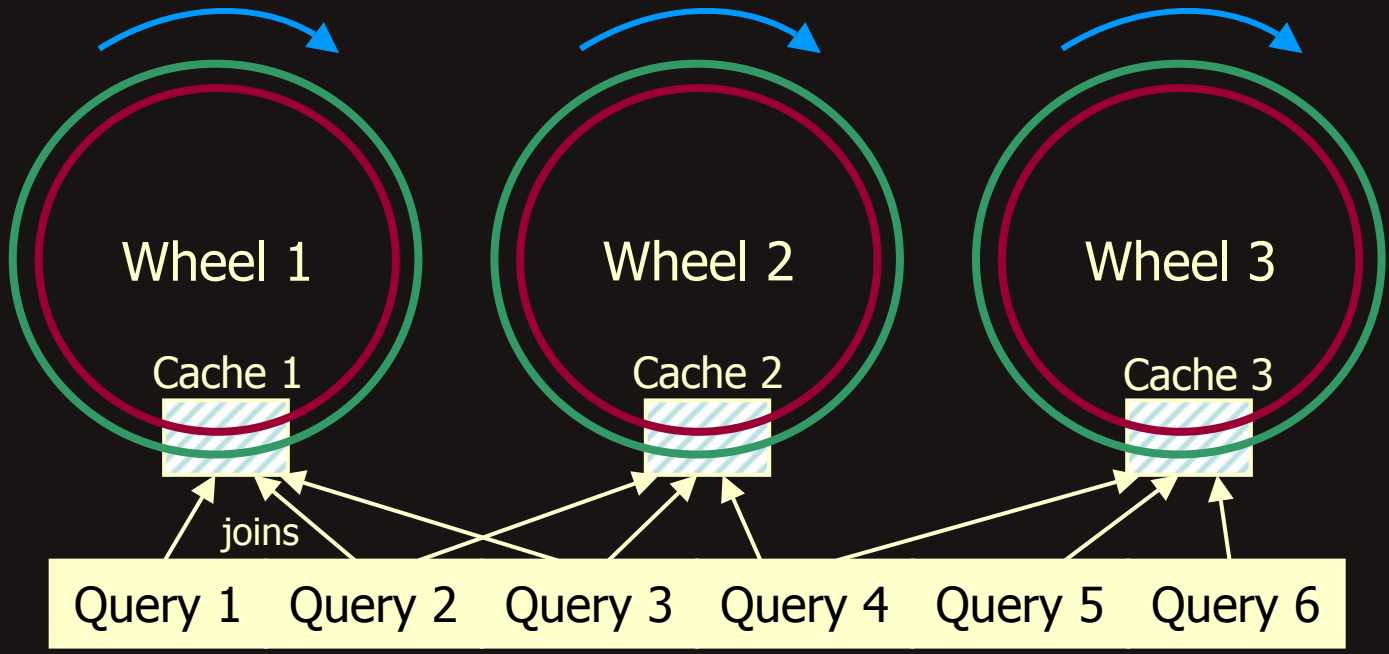
SQL Server	TAM
Resolve Target of 66 deg² requires:	Resolve Target of 66 deg² requires:
Step A: Find Candidates	*Step A: Find Candidates*
– Input data = 108 MB covering 104 deg²	– Input data = 16 MB /field (buffer included)
– Time= ~ 6 h	– Time= 885s x 66x 4 = 65h
Step B: Find Clusters	*Step B: Find Clusters*
– Input Data = 1.5 MB	– Input Data = ???
– Time = 20 minutes	– Time = 120s x 66 x 4 = 9h
Total time = 6h 20'	Total Time for 1 CPU = 74 h
	Using 10 nodes on Grid ~ 7 hours

SQL version is at least 10x faster, but more precise number not possible because SQL version uses 0.001 redshift, whereas TAM used 0.01. Also, SQL machine is 2.6GHz, TAM machine was 600Mhz.

Megajoins & Ferris Wheel

Megajoins are joins of *megastreams* (Grossman et al. 2002).

- Merging and filtering of high volume data streams



The data Ferris Wheel concept. Sequential scans of the data are repeated indefinitely, with each query operating on the data in the cache at that moment. Multiple wheels can provide parallelism and less idle time per query.

JHU IVDGL cluster

- JHU IVDGL Cluster
  - 20-node dual-CPU LINUX compute cluster running Condor
  - 10-node SQL server data cluster will be available soon with SDSS data on it
- Cluster status

Usage Stats

Application	*#jobs*
ivdgl	2605
sdss	5815
usatlas1	4
uscms01	920
gyuri	44
TOTAL	9388

VO logging

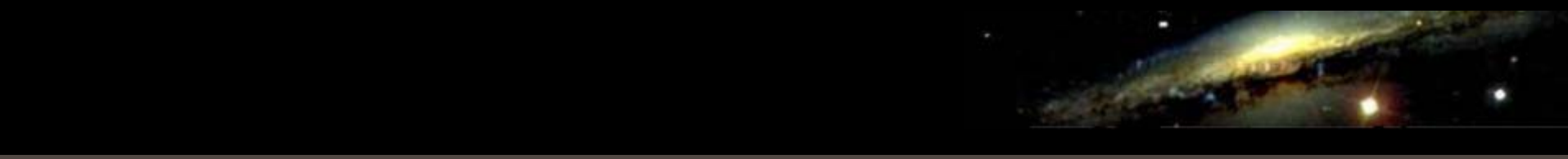
- What should we be logging?
  - Web hits, queries, SOAP requests, …
- Data model for logging
  - WebLog table
  - LogSource table
  - Do we need separate table for each type of request?
- Web services and clients for log harvesting
  - Each archive has web service for reading logs
    - Outputs logs in VOTable(?) format
  - VO has web client(s) for log harvesting
  - VO has web service to display traffic

VO WebLog Schema

- Date – Date request submitted
- ClientIP – IP Address request came from
- Request – The actual HTTP request (URL+params)
- Event – Type of event – query, webpage, etc.
  - Should Method be a separate field (GET,POST, etc.)?
- TimeIn – Time request was received
- TimeOut – Time request was serviced
- Volume – Size in bytes of the response
- Status – 0 for pending, 1 for done, 2 for failed
- Response – Error message if failed, or response
- Query – actual query, if applicable

VO Logging Issues

- VO should define minimum log schema
- Archives may log more data locally
  - Optionally make these available to VO
- VO should provide WS to export logs (?)
- Need to meet privacy concerns
  - Level of visibility configurable
  - VO should at least collect aggregates
    - Web hits, total bytes in queries, volume of results etc.
- Grid community should have input on this



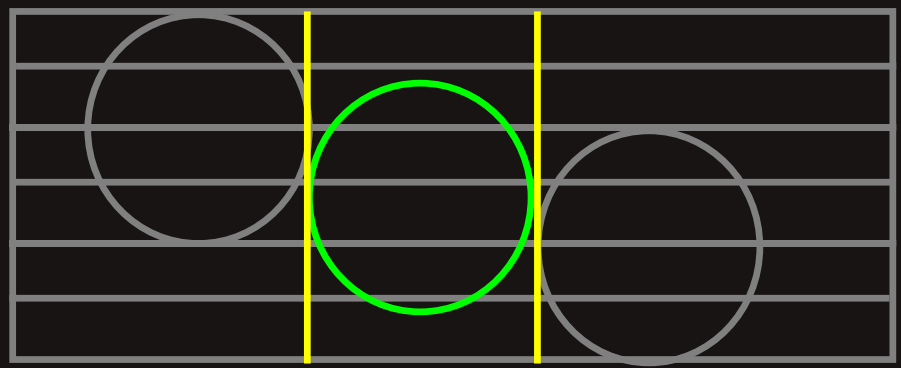
Partitioning Process

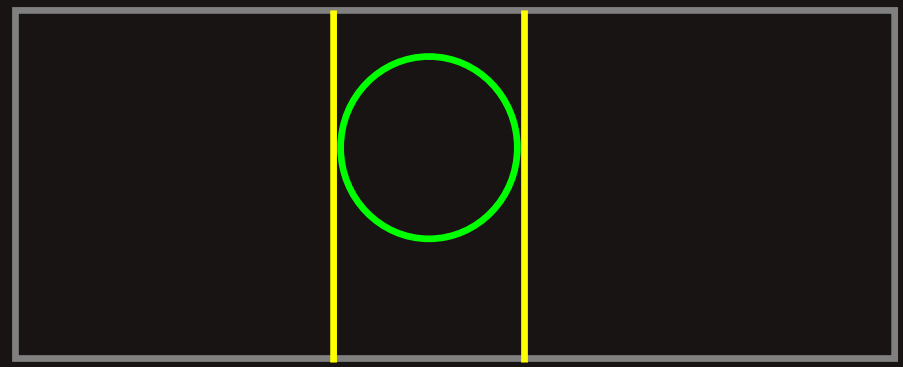
- Generate partitions (n_server, n_buffers)
  1. Calculate the number of objects included on each Zone, N_z.
  2. Compute the accumulated distribution of objects , A, for each Zone. A_{Zi} = Sum (N_{zi}), i = 1 .. i
  3. Assign the 100/n_servers % of objects to each server.
  4. Add to each server the buffer zones.
  5. Add wrap objects.

  Output:
  - ServerZones (ServerId, ZoneID, objID, ra, dec, x, y, z, wrap, native, …) Indexed by ZoneID and objID for fast access!
  - Servers (ServerID, nObj, minZoneID, maxZoneID, overlapMinZoneID, overlapMaxZoneID, minDEC, maxDEC)
- Transfer data from main server to cluster members.

Neighborhoods best Performance

- Building Neighborhoods performs best when the zone height is equal to the radius of the neighborhood.



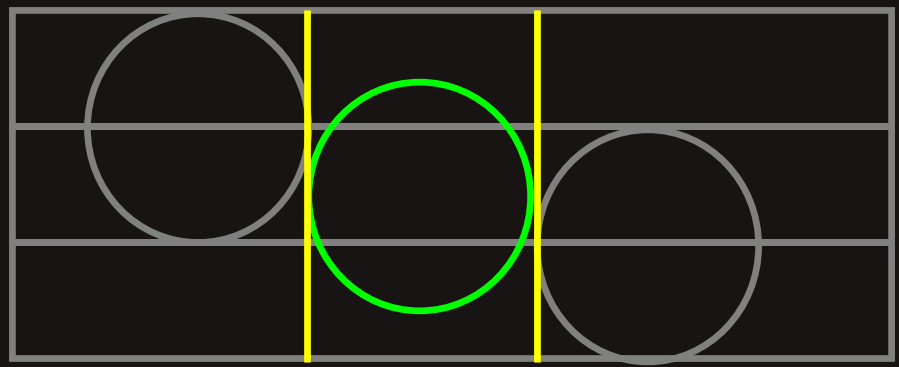


small radius imply joins with two or more northern zones and two or more southern neighbors.

Tall zones require many more pairs and the work rise quadraticly.

Neighborhoods best Performance

- Building Neighborhoods performs best when the zone height is equal to the radius of the neighborhood.



the center zone requires just a join with the upper and southern zones. A box of 3r x 2r.

Work to do

- Include wrap objects for ra nearby 0,360
- Generalize the algorithm to deal with empty areas. (Step 5.)
- Use the sqlLoader to describe the workflow, register and manage tasks and transactions.